Проектирование умножителя методом правого сдвига и сложения с управляющим автоматом в базисе ПЛИС

Андрей СТРОГОНОВ, д. т. н.
andreistrogonov@mail.ru
Алексей БЫСТРИЦКИЙ, к. т. н.
bystritsky@bk.ru

Для проектирования КИХ-фильтров в базисе процессоров цифровой обработки сигналов (ЦОС-процессор) используется общепринятая методика умножения с накоплением с применением так называемых МАС-блоков из-за отсутствия встроенных комбинационных умножителей [1].

Так, для КИХ-фильтра на четыре отвода необходимо четыре МАС-блока. Применение же распределенной арифметики, например, для КИХ-фильтра на четыре отвода в базисе ЦОС-процессоров позволяет использовать лишь составные части МАС-блока: это четыре блока логики генерации частичных произведений, получаемых с помощью булевой операции, логическое И к множимому (многоразрядные константы, являющиеся коэффициентами фильтра) и битовому значению множителя с выходов линии задержки и масштабирующий аккумулятор. При этом дерево многоразрядных сумматоров не сокращается. А для реализации того же фильтра в базисе ПЛИС на последовательной распределенной арифметики не требуется логики генерации частичных произведений и дерево многоразрядных сумматоров, так как эти функции выполняет таблица перекодировки (LUT), содержащая комбинацию сумм коэффициентов являющихся константами всех возможных вариантов, адресуемая всеми битами всех входных переменных. Для суммирования значений с выходов LUT используется лишь одна составная часть МАС-блока — масштабирующий аккумулятор (рис. 1). Вот почему авторы считают целесообразным продолжить рассмотрение механизмов умножения с накоплением.

В работах [2, 3] обсуждалось проектирование МАС-блока на примере умножения десятичного числа 10 на число 11 (целочисленные беззнаковые числа), а также умножение десятичного числа 10 на 11 на примере мегафункции ALTMEMMULT САПР ПЛИС Quartus II Altera. Мегафункция ALTMEMMULT предназначена для умножения числа на константу, которая хранится в блочной памяти ПЛИС (M512, M4K, M9K и MLAB-блоки), обеспечивая наивысшее быстродействие, лимитируемое латентностью в два такта. Процесс умножения с помощью мегафункции ALTMEMMULT составляет 20 синхроимпульсов с момента появления сигнала разрешения загрузки. А для процесса умножения с помощью разработанного МАС-блока необходимо было

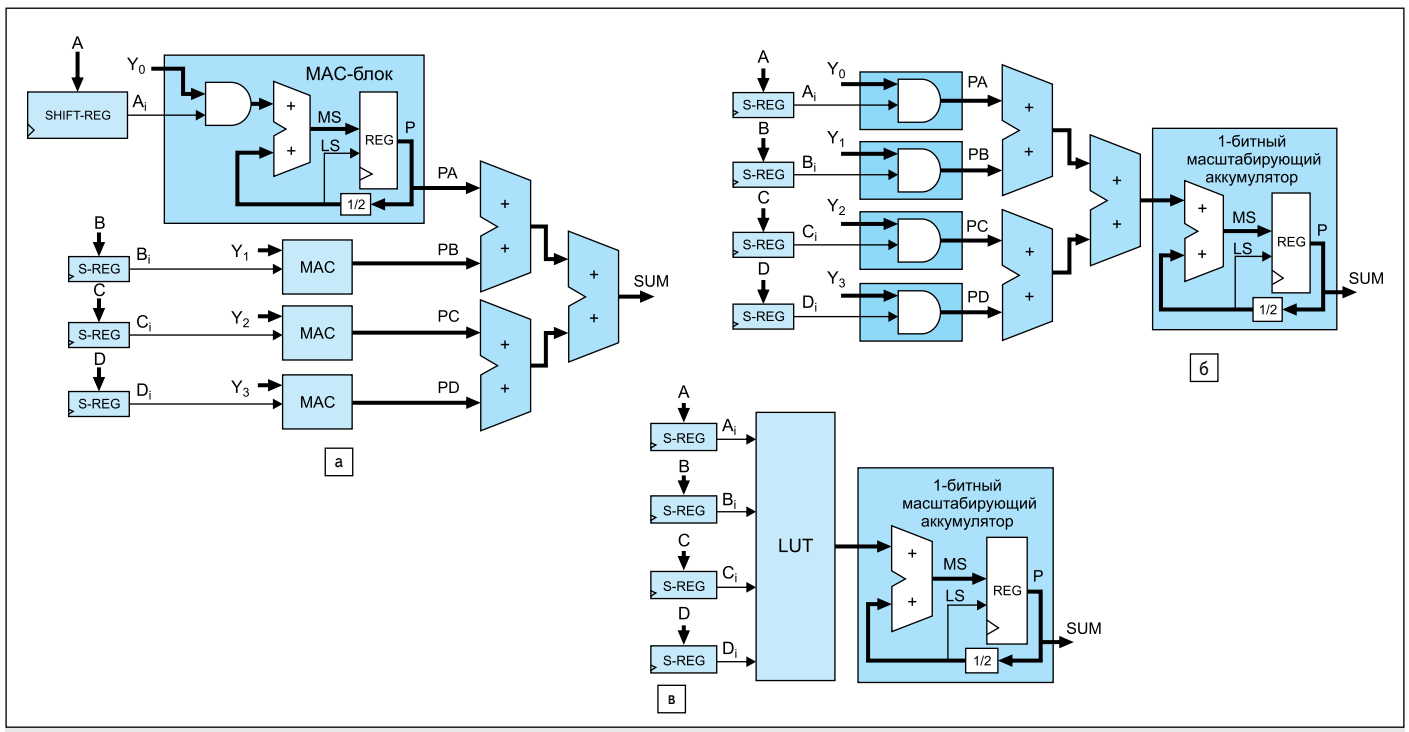

Рис. 1. Миграция проекта КИХ-фильтра на четыре отвода: а, б) реализация в базисе сигнальных процессоров; в) реализация в базисе ПЛИС

	Cout	2 тетрада	1 тетрада	2 тетрада	2 тетрада	1 и 2 тетрады
a 10 x 11			1 0 1 0 1 0 1 1			
$p^{(0)}$ $+x_0a$		0 0 0 0 1 0 1 0	0 0 0 0	0 a	0 10	
$2p^{(1)}$ $p^{(1)}$ $+x_1a$	0	1 0 1 0 0 1 0 1 1 0 1 0	0 0 0 0	a a/2 a	10 5 10	80
$2p^{(2)}$ $p^{(2)}$ $+x_2a$	0	1 1 1 1 0 1 1 1 0 0 0 0	0 1 0 0 0	a/2+a (a/2+a)/2 0	15 7 0	120
$2p^{(3)}$ $p^{(3)}$ $+x_3a$	0	0 1 1 1 0 0 1 1 1 0 1 0	0 0 1 1 0 0	(a/2+a)/2 ((a/2+a)/2)/2 a	7 3 10	60
$2p^{(4)}$ $p^{(4)}$	0	1 1 0 1 0 1 1 0	1 1 0 1 1 1 0	((a/2+a)/2)/2+a (((a/2+a)/2)/2+a)/2	13 6	110

Рис. 2. Принцип умножения методом правого сдвига и сложения. Умножение десятичного числа 10 на десятичное число 11

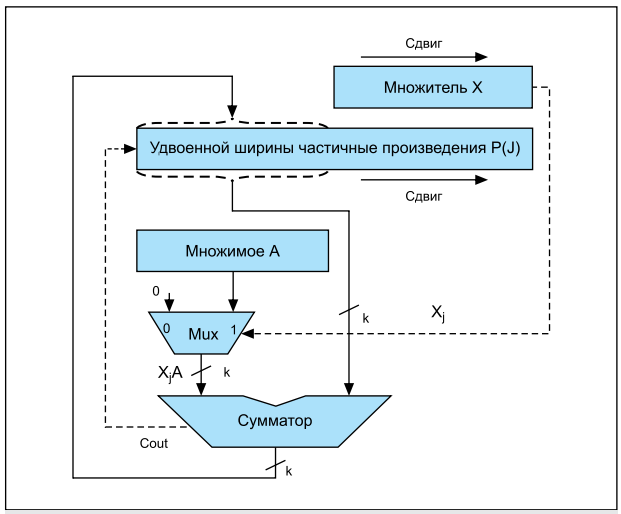

Рис. 3. Структурная схема умножителя методом правого сдвига и сложения

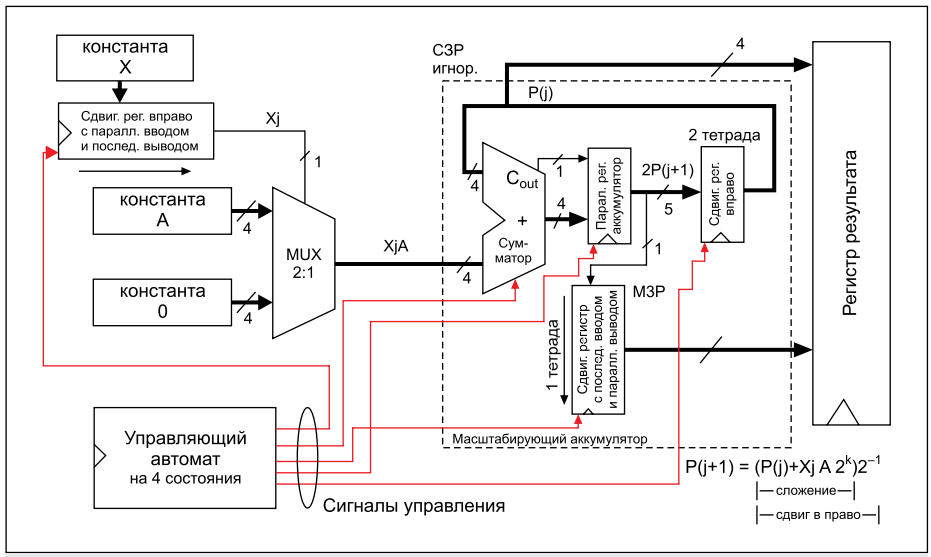

Рис. 4. Предлагаемая структурная схема умножителя с управляющим автоматом

18 синхроимпульсов [3]. Однако недостатком разработанной схемы являлось отсутствие управляющего автомата, и необходимые сигналы управления приходилось формировать вручную непосредственно в векторном редакторе.

Встраивание автомата в схему позволяет получить готовую функцию без использования дополнительных управляющих сигналов для умножения двух четырехразрядных чисел без знака. На входах MAC-блока потребуется всего лишь три сигнала: сигнал асинхронного сброса res, сигнал тактирования clk и сигнал разрешения загрузки числа X (множителя) в сдвиговый регистр load_PSC. Для корректной работы схемы необходимо обнулить все регистры умножителя (активный — высокий уровень сигнала res). Поскольку все регистры, в том числе и регистр для хранения состояний в управляющем автомате, обнуляются единожды и только перед началом работы, то для упрощения процесса разработки схемы можно воспользоваться асинхронным сбросом.

Идея схемы метода умножения методом правого сдвига с накоплением была заимствована из [4–6]. Такая схема в адаптированном варианте представлена на рис. 2. В [4–5] высказана идея реализации этого метода (рис. 3).

На рис. 4 предложена структурная схема метода умножения на основе управляющего автомата.

Структурная схема умножителя двух 4-разрядных чисел, представленных в двоичном коде (целые, положительные числа), состоит из шинного мультиплексора «2 в 1», сдвигового регистра, цифрового автомата на четыре состояния и масштабирующего аккумулятора (рис. 5). Шинный мультиплексор «2 в 1» и сдвиговый регистр вправо реализуют логику генерации частичных произведений. В основе масштабирующего аккумулятора лежит синхронизируемый сумматор с сигналом разрешения тактирования (рис. 6).

Приведем пример кода языка VHDL управляющего автомата на четыре состояния:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
ENTITY avtomat IS
    PORT(
    Res,clk                          : IN     STD_LOGIC;
    Ena_Add, LoadPSC,ena_shift,Stop  : OUT    STD_LOGIC;
    Qa : OUT STD_LOGIC_VECTOR(4 downto 0));
END avtomat;
ARCHITECTURE a OF avtomat IS
TYPE state_values IS (SA, SB, SC, SD);
signal state,next_state: state_values;
SIGNAL cnt: STD_LOGIC_VECTOR(4 downto 0);
BEGIN
statereg: process(clk,Res)
begin
  if (Res = '1') then state<=SA;
    elsif (clk'event and clk='1') then
      state<=next_state;
  end if;
end process statereg;
process(state)
begin
case state is
when SA=> next_state<=SB;
when SB=> next_state<=SC;
when SC=> next_state<=SD;
when SD=> next_state<=SA;
end case;
end process;
process (state)
begin
case state is
when SA=>Ena_Add<='0';
        LoadPSC<='0'; ena_shift<='1';
when SB=>
        Ena_Add<='1'; LoadPSC<='0';
        ena_shift<='0';
when SC=>
        Ena_Add<='0';
        LoadPSC<='0';
        ena_shift<='0';
when SD=>
        Ena_Add<='0';
        LoadPSC<='1';
        ena_shift<='1';
end case;
end process;
process (clk, res)
  begin
    if (res = '1') then
      cnt <=(others=>'0');
    elsif (clk'event and clk = '1') then
      if cnt = "10001" then Stop <= '1';
      else cnt <= cnt+'1';
      end if;
    end if;
end process;
      Qa <= cnt;
END a;
```

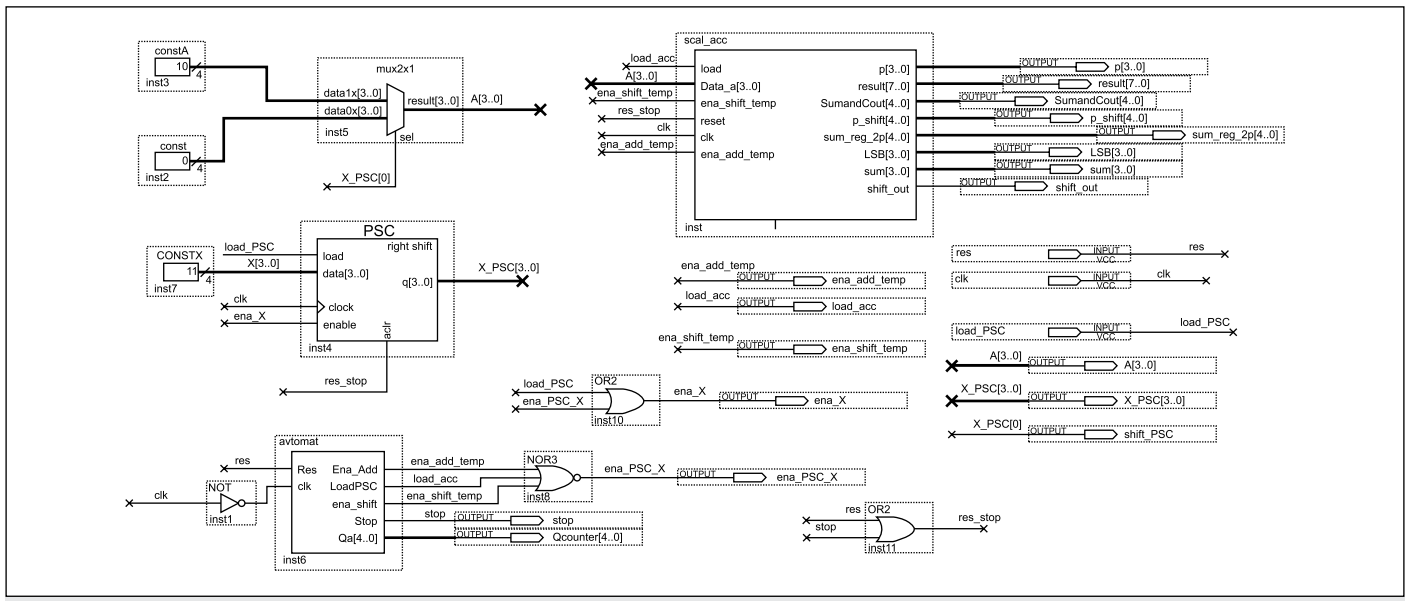

Рис. 5. Схема умножителя в САПР ПЛИС Quatus II

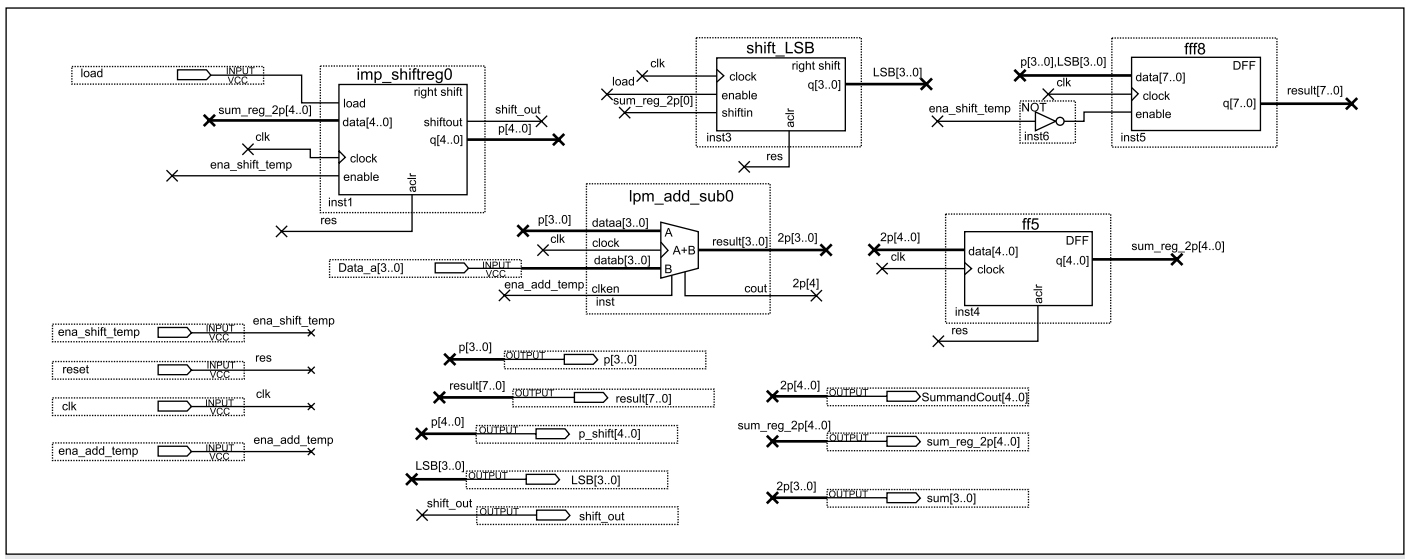

Рис. 6. Схема масштабирующего аккумулятора

Цифровой автомат представляет собой автомат Мура на четыре состояния: SA (1-е состояние), SB (2-е состояние), SC (3-е состояние) и SD (4-е состояние). Он формирует три управляющих сигнала: ena_add_temp, load_acc и ena_shift_temp по срезу фронта синхроимпульса clk (пример), два из которых (ena_add_temp и load_acc) — неперекрывающиеся (рис. 7). По активному уровню сигнала асинхронного сброса res автомат попадает в первое состояние SA, а по низкому уровню res осуществляется переход по состояниям. На рис. 7 над сигналами ena_shift_temp и load_acc нанесены номера состояний цифрового автомата. Так, сигнал ena_shift_temp активен в первом и четвертом состояниях.

Сигнал ena_add_temp формируется, когда автомат находится во втором состоянии SB. Сигнал load_acc формируется в четвертом состоянии. Таким образом, удается обеспечить конвейерный режим работы масштабирующего аккумулятора, при этом формируемые сигналы ena_add_temp, load_acc и ena_shift_temp синхронны для всех регистров умножителя. Так, второй передний фронт синхроимпульса оказывается ровно над половиной высокого уровня сигнал ena_add_temp, что обеспечивает корректную работу синхронного сумматора (рис. 7). Четвертый передний фронт также оказывается ровно над половиной высоких уровней сигналов load_acc и ena_shift_temp, что обеспечивает загрузку числа с промежуточного регистра (аккумулятора) в сдвиговый регистр lpm_shiftreg0. Пятый фронт также оказывается ровно над серединой высокого уровня сигнала ena_shift_temp для выполнения операции деления на 2 (сдвиг вправо).

В управляющий автомат встроен суммирующий счетчик, который подсчитывает число синхроимпульсов. И при достижении 18-го синхроимпульса (отсчет ведется с нуля) вырабатывается сигнал останова работы умножителя, который сбрасывает все регистры умножителя в ноль, кроме регистра результата (fff8), запись информации в который осуществляется при низком уровне сигнала ena_shift_temp.

На информационные входы мультиплексора подключаются две константы: множимое (число A) и логический ноль, а на адресный вход мультиплексора — младший разряд (множителя) сдвигового регистра (мегафункция LPM_SHIFTREG) с параллельным вводом информации и последовательным выводом. Такой регистр будем называть конвертером (это преобразователь параллельного кода в последовательный) и обозначать как PSC. Регистр настроен на сдвиг вправо и имеет синхронные сигналы загруз-

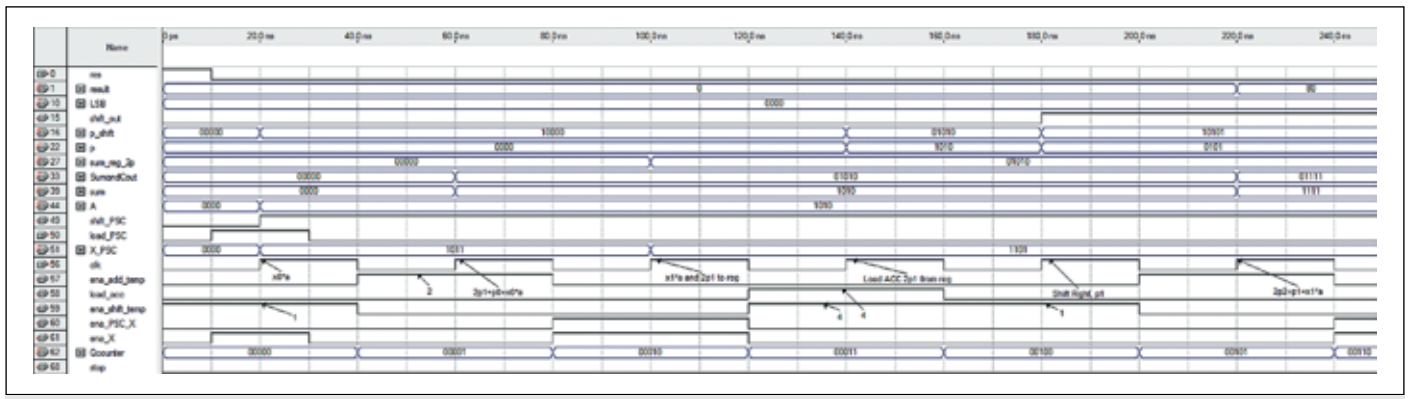

Рис. 7. Временные диаграммы процесса вычисления удвоенных первого 2P(1) и второго частичных произведений 2P(2)

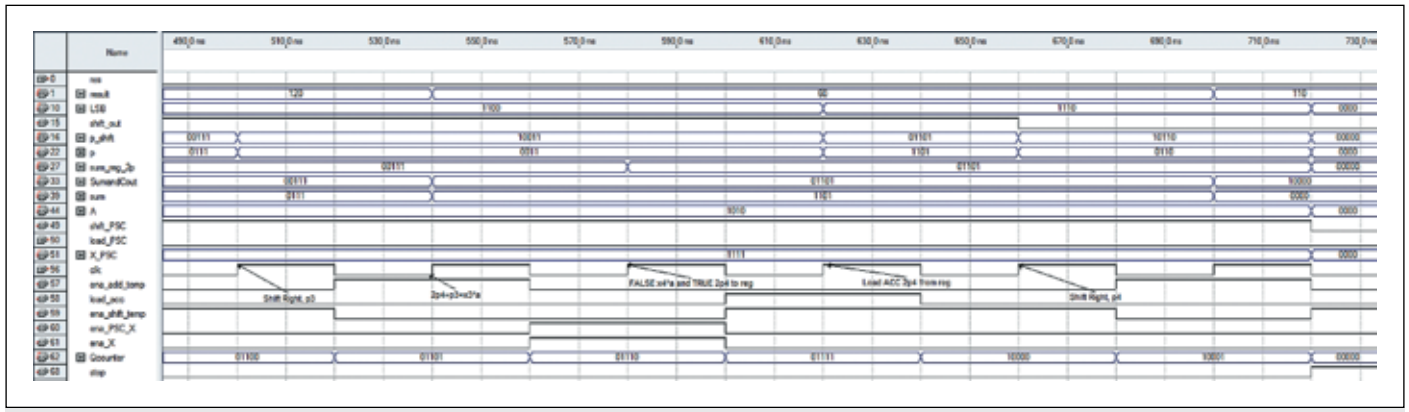

Рис. 8. Временные диаграммы процесса вычисления удвоенного четвертого частичного произведения 2P(4)

ки load и разрешения тактирования enable, активные при высоком уровне.

При сдвиге вправо в старший разряд заносится логическая 1, а младший разряд теряется. В сдвиговый регистр по высокому уровню сигнала load_PSC записывается число X (множитель). При этом сигнал ena_X на входе enable должен быть высокого уровня, который может быть получен объединением по ИЛИ сигналов load_PSC и ena_PSC_X. Сигнал ena_PSC_X является выходом цифрового автомата и получается в результате применения операции 3 ИЛИ-НЕ над сигналами ena_add_temp, load_acc и ena_shift_temp, то есть он возникает только в том случае, когда сигналы ena_add_temp, load_acc и ena_shift_temp — низкого уровня.

В первом такте синхроимпульса сигнал ena_PSC_X не активен и не оказывает влияние на формирование сигнала ena_X. Он используется при последующем сдвиге множителя X вправо. Параллельная загрузка числа X происходит по принципу: если активен сигнал загрузки load_PSC, то, значит, должен быть активен и сигнал разрешения тактирования ena_X.

Недостатком такого решения является неразрешенный сдвиг вправо пятиразрядным сдвиговым регистром (lpm_shiftreg0) масштабирующего аккумулятора при первом фронте синхроимпульса, на выходах которого появляется 1 в старшем разряде при активном сигнале ena_shift_temp, что и показано на рис. 7. Однако это не влияет на работу масштабирующего аккумулятора, так как перед сложением сдвиговый регистр перегружается правильным значением по четвертому фронту синхросигнала при активных сигналах load_acc и ena_shift_temp. Неразрешенный сдвиг будет проявляться и при последующих синхроимпульсах при наличии активного уровня сигнала ena_shift_temp. Для противодействия этому явлению старший значащий разряд p[4] регистра lpm_shiftreg0 после сдвига просто игнорируется, а на вход сумматора поступает уменьшенное на два значение частичного произведения p[3..0] с 4-битной точностью представления.

Рассмотрим работу масштабирующего аккумулятора. По первому фронту синхроимпульса clk при активных сигналах load_PSC и ena_X происходит загрузка числа X в сдвиговый регистр. По второму фронту синхроимпульса управляющий автомат вырабатывает синхронный сигнал разрешения тактирования ena_add_temp для сумматора масштабирующего аккумулятора. На выходах сумматора формируется первое удвоенное частичное произведение 2P(1) = P(0)+X(0)*A (шина 2p[3..0]), равное 10, при этом P(0) = 0. Выход переноса Cout и 2P[3..0] объединяются в пятиразрядную шину, значения которой сохраняются в промежуточном параллельном регистре-аккумуляторе (выход sum_reg_2p[4..0]) по третьему фронту синхроимпульса.

По четвертому фронту синхроимпульса при активных сигналах load_acc и ena_shift_temp происходит загрузка первого удвоенного частичного произведения, равного 10, в сдвиговый регистр. По пятому фронту при активном сигнале ena_shift_temp происходит сдвиг вправо удвоенного частичного произведения 2P(1). При этом на выходах сдвигового регистра образуется число 5. По шестому фронту синхроимпульса при активном сигнале ena_add_temp произойдет сложение числа A (десятичное число 10) с числом 5, и на выходах сумматора сформируется второе частичное произведение 2P(2) = P(1)+X(1)*A, равное числу 15, что в точности соответствует принципу умножения, продемонстрированному на рис. 2.

На рис. 8 показан момент окончания счета. По 15-му фронту синхроимпульса (отсчет по тексту ведется с первого фронта синхроимпульса) произойдет неразрешенная загрузка четвертого бита в сдвиговый регистр PSC по высокому уровню сигнала ena_PSC_X и сформируется еще одна копия сигнала A, то есть X(4)*A. Однако это тоже не повлияет на результат, так как последующего сложения уже не будет. Потребуется еще два такта синхроимпульса для загрузки удвоенного произведения 2P(4) в аккумулятор и последующий сдвиг вправо для формирования P(4).

	Cout	2 тетрада	1 тетрада	1 и 2 тетрады
a x		1 1 1 1 1 1 1 1		
$p^{(0)}$ $+x_0a$		0 0 0 0 1 1 1 1		
$2p^{(1)}$ $p^{(1)}$ $+x_1a$	0	1 1 1 1 0 1 1 1 1 1 1 1	1 0 0 0	120
$2p^{(2)}$ $p^{(2)}$ $+x_2a$	1	0 1 1 0 1 0 1 1 1 1 1 1	0 1 0 0	180
$2p^{(3)}$ $p^{(3)}$ $+x_3a$	1	1 0 1 0 1 1 0 1 1 1 1 1	0 0 1 0	210
$2p^{(4)}$ $p^{(4)}$	1	1 1 0 0 1 1 1 0	0 0 0 1	225

Рис. 9. Принцип умножения методом правого сдвига и сложения. Умножение десятичного числа 15 на десятичное число 15

И по 18-му такту импульса результат умножения будет доступен в регистре результата fff8.

Было проведено тестирование разработанной схемы на предмет умножения двух 4-разрядных чисел без знака в диапазоне входных значений от 0 до 15. На рис. 9 показан принцип умножения десятичного числа 15 на 15, а на рис. 10 — временные диаграммы процесса умножения.

Выводы

Разработан МАС-блок для умножения двух 4-разрядных чисел без знака с использованием метода умножения и накопления. Управление блоком осуществляется с помощью цифрового автомата на четыре состояния. Предложенная схема реализации МАС-блока в базисе ПЛИС может быть использована при проектировании КИХ-фильтров. ■

Литература

1. Goslin G. R. A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance. V. 1.0. 1995 — www.xilinx.com
2. Строгонов А., Быстрицкий А. КИХ-фильтр на распределенной арифметике: проектируем сами // Компоненты и технологии. 2013. № 3.
3. Строгонов А., Быстрицкий А. Проектирование параллельных КИХ-фильтров в базисе ПЛИС // Компоненты и технологии. 2013. № 6.
4. Computer Arithmetic: Algorithms and Hardware Designs. Oxford U. Press, 2nd ed., 2010.
5. Khadivi P. Car_multiply.pdf. Isfahan University of Technology — http://khadivi.iut.ac.ir
6. Koren I. Digital Computer Arithmetic. ECE 666. Part 3. Sequential Algorithms for Multiplication and Division. University of Massachusetts Dept. of Electrical & Computer Engineering — http://euler.ecs.umass.edu/arith/parts-pdf/Part3-seq.pdf

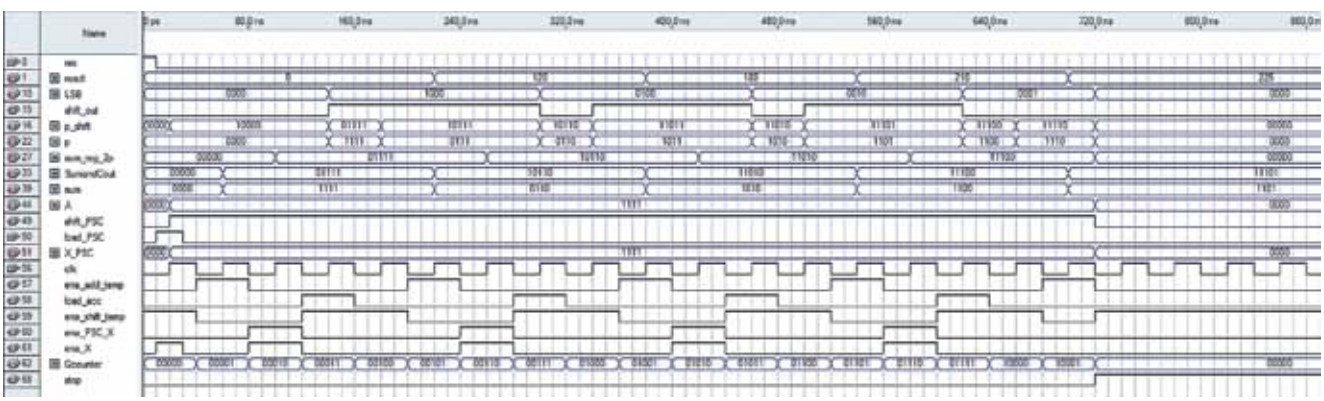

Рис. 10. Умножение десятичного числа 15 на десятичное число 15. Результат — 225